Greedy 2-Approximation Algorithm for Negative Due Dates

For a subset of jobs $S \subseteq \{1, \ldots, n\}$ let:

$$r(S) := \min_{j \in S} r_j \qquad p(S) := \sum_{j \in S} p_j \qquad d(S) := \max_{j \in S} d_j$$

Lemma 2.3.

Let $L^*_{\max}$ denote the optimal value. For each subset S of jobs

$$L^*_{\max} \geq r(S) + p(S) - d(S) \ .$$

Proof:... □

Algorithm EDD (earliest due date): Whenever the machine is idle, start to process among all available jobs the one with the earliest due date.

Theorem 2.4.

For the case of non-positive due dates $d_j \leq 0$ for all jobs j, Algorithm EDD is a 2-approximation algorithm.

Proof:... □

The k-Center Problem

Given: A finite metric space V with distances d_{ij} for $i, j \in V$ and $k \in \mathbb{N}$.

Task: Find k centers in V, i. e., $S \subseteq V$ with $|S| = k$.

Objective: Minimize $\max_{i \in V} d(i, S)$ where $d(i, S) := \min_{j \in S} d_{ij}$.

Greedy Algorithm:

1. pick arbitrary $i \in V$ and set $S := \{i\}$;
2. while $|S| < k$ let $j := \arg\max_{j \in V} d(j, S)$ and set $S := S \cup \{j\}$;

Theorem 2.5.

The algorithm is a 2-approximation algorithm for the k-Center Problem.

Proof:... □

Theorem 2.6.

There is no α-approximation algorithm for the k-center problem for $\alpha < 2$, unless $P = NP$.

Proof: Reduction from Dominating Set Problem... □

Scheduling Jobs on Identical Parallel Machines

Given: n jobs $j = 1, \ldots, n$ with processing time $p_j \geq 0$, $j = 1, \ldots, n$, and m identical parallel machines.

Task: Process each job j nonpreemptively for p_j units of time on one of the m machines. A machine can process at most one job at a time.

Objective: Minimize the maximum machine load, i. e., the maximum completion time $C_{\max} := \max_{j=1,\ldots,n} C_j$ (makespan).

Theorem 2.7.

This scheduling problem is strongly *NP*-hard.

Proof: Reduction from 3-Partition. . . □

Local Search

Local Search Algorithm

1. start with an arbitrary schedule;
2. let $j := \arg\max_{j=1,\ldots,n} C_j$;
3. if there is a machine i with load $< C_j - p_j$, reassign j to i and goto 2;

Theorem 2.8.

When the algorithm terminates, the makespan of the final solution is at most twice the optimum makespan.

Proof:. . . □

Lemma 2.9.

If the Local Search Algorithm always moves job j to a currently least loaded machine, it terminates after at most n iterations.

Proof:. . . □